NATURAL

# Natural

## Debugging Manual

## Version 5.1.1 for UNIX and OpenVMS

software AG

This document applies to Natural Version 5.1.1 for UNIX/OpenVMS and to all subsequent releases. Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

# Table of Contents



# Natural Debugger

This document applies to Natural Version 5.1.1 for UNIX and OpenVMS, and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes and new editions.

- General Information
- Local and Remote Debugging
- Using the Debugger

# General Information

This documentation contains information related to Natural under UNIX and OpenVMS.

- About This Documentation
- Other Natural Documentation

# About This Documentation

This documentation provides information on how to debug Natural applications. You can do this regardless of whether the application is located on a UNIX, OpenVMS or Windows computer, or whether the application is distributed, with access to Natural DCOM or Natural RPC servers.

If your Natural development environment does not provide a debugger, which is the case for Natural for UNIX and for OpenVMS, you can install the Natural Debugger separately on a Windows computer and call this debugger remotely from Natural with the DEBUG system command. Alternatively, you can use the Natural remote debugger from within Natural for Windows NT Version 5.1.1. The following Natural versions can connect to a remote debugging session:

- Natural for UNIX Version 5.1.1
- Natural for OpenVMS Version 5.1.1
- Natural for Windows NT Version 5.1.1

This documentation contains the following sections:

- Section 1, "Local and Remote Debugging", describes the difference between local and remote debugging, explains the scenarios in which remote debugging can be used, and describes how to prepare an environment for remote debugging.
- Section 2, "Using the Debugger", explains how you can use the Natural Debugger with a given application.

# Other Natural Documentation

For an overview other Natural documentation available for UNIX and OpenVMS, please refer to the User's Guide for UNIX and OpenVMS.

# Local and Remote Debugging

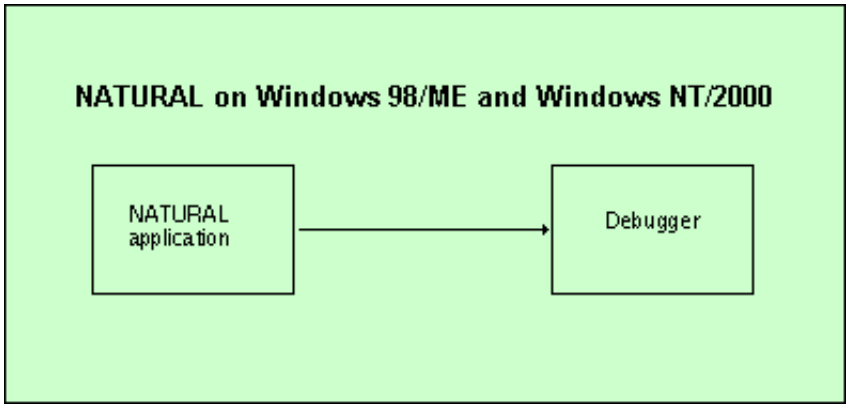
The following topics are covered below:

- Local Debugging
- Remote Debugging
- Setting Up Your Environment
- Running a Natural Remote Debugging Session

---

## Local Debugging

Local debugging, as opposed to remote debugging, is done with a debugger integrated in Windows. This debugger can be used to debug Natural applications running on the same computer.



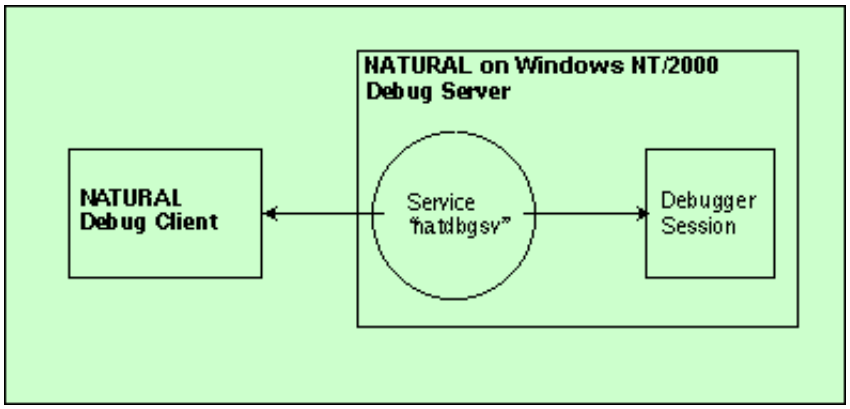

## Remote Debugging

There are several scenarios of how you can use remote debugging: a single Natural client runs under the control of one remote debugging session or a distributed Natural application runs under the control of several remote debugging sessions. Such a distributed application may include both Natural RPC and DCOM servers or even components not written in Natural, such as Visual Basic clients.

- Scenario 1: Debugging a Single Natural Application
- Scenario 2: Debugging a Distributed Natural Application
- Scenario 3: Debugging The Natural Part of a Heterogeneous Application

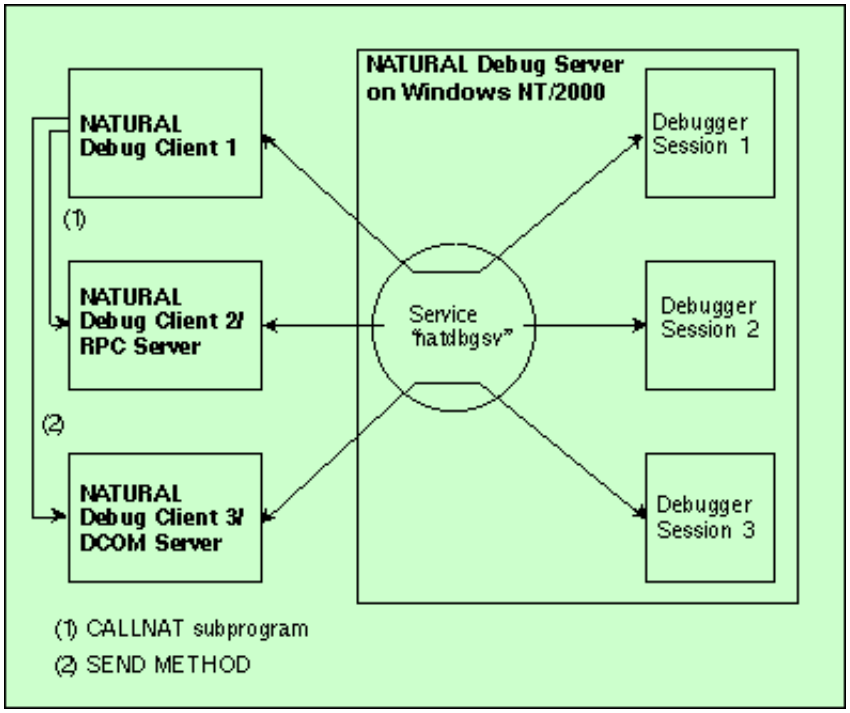
## Scenario 1: Debugging a Single Natural Application




## Scenario 2: Debugging a Distributed Natural Application

To debug each component of the following distributed Natural application, you enter "DEBUG *objectname*" in the command line of Natural Debug Client 1. The first time the Natural Debug Client calls a subprogram on a Natural RPC server, for example, a new debug session is opened for the RPC server. Then, the RPC server's processing is debugged. The debug session is closed as soon as the RPC server is terminated.

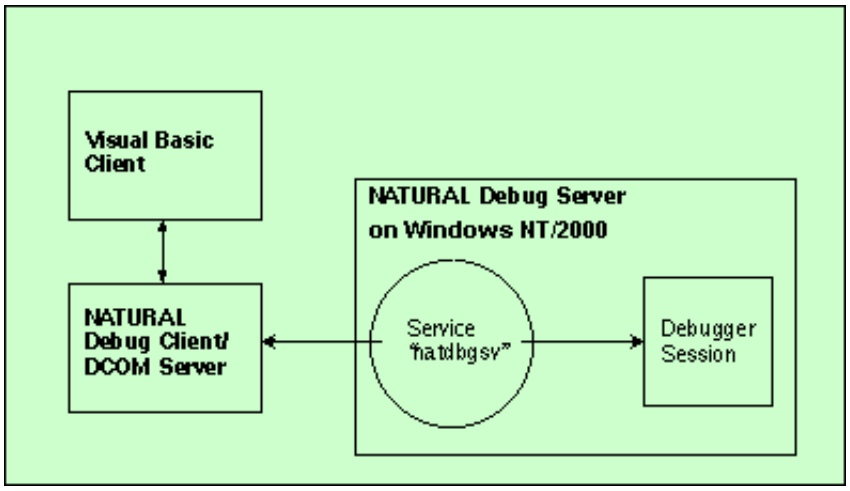
The same applies to a Natural DCOM server.



# Scenario 3: Debugging The Natural Part of a Heterogeneous Application

As in Scenario 2, the first time a method on the DCOM server is called, a new debug session is opened for the DCOM server, the DCOM server's processing is debugged, and the debugger session is closed as soon as the DCOM server is terminated:



# Setting Up Your Environment

## Windows NT/2000 Side

Install the Natural remote debugger or install Natural for Windows NT 4 PL 6 or Windows 2000. This also installs the Natural Remote Debugging Service.

To uninstall the Natural Remote Debugging Service, enter "natdbgsv -u" in the command line. To view the current service's port name and version, enter "natdbgsv -s". To re-install the service on a different port, uninstall it first and then enter "natdbgsv -i *portnumber*", where *portnumber* is the value of the RDPORT profile parameter.

**Note:**
If you install the Natural Remote Debugging Service on a port other than 2500, you have to change the value of the RDPORT profile parameter accordingly on the UNIX client computer.

## Natural Side

Start Natural with the following three profile parameter settings:

- RDACTIVE=ON;
- RDNODE=*nodename*, where *nodename* is the name of the Windows NT/2000 server;
- RDPORT=2500 (or another port number, depending on which port you have installed the Natural Remote Debugging Service on the Windows NT computer).

# Running a Natural Remote Debugging Session

To run a Natural Remote Debugging Session:

1. Ensure that the Windows NT/2000 server has been started and that the service "natdbgsv" is active.
2. Enter "DEBUG *objectname*" in the Natural command line, where *objectname* is the name of the Natural object you wish to debug.

The Natural Debuggerbecomes active on the Windows NT/2000 computer, with the following window caption: "Debugging remote Natural client (*nodename*::*username*::*process-id*)", where *nodename* is the name of the computer where Natural is running, *username* is the name of the Natural user and *process-id* is the Natural process ID.

# Using the Debugger

The following topics are covered below:



---

## General Information

The Natural Debugger provides application developers with various benefits and helpful features. For example, the Natural Debugger enables you to:

- gain a faster understanding of programs written by another developer when taking over an application to maintain, improve or expand it;
- in general, develop your own applications more quickly;
- better understand the logic of Natural if you are a first-time user.

This can be achieved by:

- temporarily controlling or influencing the program flow of a Natural application by modifying variables;
- checking the program flow through a calls history indicating which objects are called from which objects;
- easily detecting logical application errors in a Natural program by checking the contents of its variables using breakpoints or conditions for program interruption;
- permanently watching variables.

## Preparing Natural Objects

To exploit the full functional scope of the Natural Debugger, you must specify the following Natural profile parameter either dynamically or in your Natural parameter file:

SYMGEN =ON

When an object is CATALOGed or STOWed and SYMGEN is set to ON, a symbol table is generated as part of the generated program. Since this table contains the information relevant to the variables active for this object, variables cannot be accessed without SYMGEN being specified, although it is still possible to debug the object.

## Starting the Natural Debugger

You can start the Natural Debugger in either of the following ways:

- Enter the command DEBUG in the Natural command line along with the name of a Natural object contained in the current library.
- For Windows: Select a Natural object from the library listing and either choose "Debugger" from the

"Tools" menu or click the corresponding button in the Natural tool bar.

**Note:**
The Natural Debugger can be applied to stowed or cataloged Natural programs and dialogs only.

Once the Natural Debugger has been started, the application frame window appears, showing additional information on the debugging session:

- Local Debugger: "Natural Debugger"
- Remote Debugger: "Debugging remote Natural client (*nodename*::*username*::*process-id*)"
- Natural application being debugged currently has control: "[waiting]"
- Debugger has control: "[break]"

The frame window contains a child window with a source listing of the specified object.

The debugger's source window is one of the five main facilities of the debugger. The remaining four main facilities are:

| Facility | Function |
|---|---|
| **Watch Variables** | Watch the contents of variables. |
| **Variable Facility** | Maintain variables. |
| **Watchpoint Maintenance** | Set and maintain watchpoints. |
| **Breakpoint Maintenance** | Set and maintain breakpoints. |

All these facilities can be invoked by restoring the corresponding icon in the "Natural Debugger" application window; they can also be chosen from the "Window" menu of the debugger menu bar.

The individual facilities are described in the remainder of this section.

# Leaving the Natural Debugger

You can leave the Natural Debugger from any point within an application by choosing either "Exit" (see below) or the corresponding tool-bar button. You can also leave the Natural Debugger by choosing "Close" from the "Control" menu of the debugger's main window.

The debugger is also terminated if the application ends without an error; the trace bar is then placed on the source code line last executed.

In the case of an error, the corresponding source is displayed in the source window and the trace bar is placed on the line which caused the error. A message window appears with the appropriate error message and a choice to either continue or end the debugging session.

Continuing the debugging session may be useful if, for example:

- your application contains any error processing (including error transactions);
- you want to display any variables before you end your debugging session.

When you leave the Natural Debugger, your window and tool bar settings are automatically saved and will be the same when you invoke the debugger again.

## "Exit" Function

"Exit" terminates the debugging session and returns control to Natural. The "Exit" function is available on the first menu in the main window of each of the five Natural Debugger main facilities.

# How to Use the Natural Debugger

Before going into detail about the debugger's source window and other main facilities, this section provides you with general information on the Natural Debugger.

- the use and handling of windows, menus and tool bars,
- the various ways to execute a function,
- the purpose of breakpoints and watchpoints,
- how to restart a debugging session.

## Windows and Menus

The Natural Debugger provides various windows and menus. As you activate different windows to access menu items, the menu bar changes to provide the functions applicable to the current item. The menu bar always contains the menus that apply to the type of window you are using.

Menu items which are assumed to be used very often are also available as tool-bar buttons in the corresponding tool bar.

Some menus are specific to the active window, some are the same for every window. Within the Natural Debugger, the "Options" menu, the "Window" menu and the "Help" menu are always part of the menu bar.

There are several ways to open and close a window. Instead of using the menus, you can click buttons from the tool bar or use accelerator keys.

In contrast to Natural itself:

- the debugger has no command line.
- the debugger's "Options" menu contains the options:
  - "Fonts", which allows you to modify the font of the currently active window;
  - "Warning messages", which allows you to decide whether warning messages on missing source code or symbolic information are to be displayed or not. A message that warns you of source code that is newer than the corresponding generated program is also affected.

## Tool Bars and Tool-Bar Buttons

Tool bars are groups of tool-bar buttons. Like the menu bar, the tool bar changes when you activate a different type of window. The Natural Debugger provides five tool bars, each containing a set of tool-bar buttons that apply to a particular type of window. By selecting "Customize toolbar" from the "Options" menu, you can change the location of the tool bar and turn it on or off.

To display a short description of the function of a tool-bar button, place the mouse pointer on the corresponding button. The description appears in the status line at the bottom of your desktop. If the function of a tool-bar button is not available in a particular window, the button is disabled.

# Special Key Combinations

A further possibility to execute a debugger function is by entering a corresponding key combination on your keyboard. Two different kinds of key combinations can be entered:

- Mnemonic combinations: Application icons, menu names, menu items and options in dialog boxes contain an underscored character (mnemonic). You can, for example, open a menu or choose a menu item by holding down ALT and pressing the mnemonic character.
- Accelerator combinations: These combinations are available only for functions that are assumed to be used very often. The valid combination (CTRL + *key*) is then displayed next to the corresponding menu item.

## Breakpoints and Watchpoints

Two types of entries can be defined in a program for debugging purposes: breakpoints and watchpoints. Each breakpoint or watchpoint is assigned a number in ascending order. The numbers are displayed in the Breakpoint or Watchpoint Maintenance window respectively.To watchpoints, a name is also assigned, which corresponds to the name of the variable they belong to.

Each breakpoint or watchpoint has a current state; possible values are "active" and "pending". While debugging a program, this state can be changed; that is, the breakpoint or watchpoint can be activated or deactivated. If its current state is "pending", a breakpoint or watchpoint is not executed and its event count is not increased either.

Every breakpoint or watchpoint has an event count, which increases every time the debug entry is passed. The number of executions of a debug entry, however, can be restricted in two ways:

- A number of skips can be specified before the breakpoint or watchpoint is executed. The debug entry is then ignored until the event count is higher than the number of skips specified.
- A maximum number of executions can be specified, so that the breakpoint or watchpoint is ignored as soon as the event count exceeds the specified number of executions.

## Restarting the Debugging Session

When you restart your debugging session, the Natural Debugger repositions to the beginning of the application while all your current settings (for example, watchpoints or breakpoints) are kept and all counters as well as the calls history are newly initialized. Thus, restarting a debugging session is useful if want to rerun your application without having to specify again the settings relevant for debugging.

You can restart your debugging session from any point within an application by choosing either the "Restart" function or the corresponding tool-bar button. The "Restart" function is available on the first menu of the menu bar of each of the five Natural Debugger main facilities.

If you are debugging a DCOM or RPC server, the "Restart" function restarts the called method or subprogram.

# Debugger Source Window

When the Natural Debugger is invoked, it receives control of the specified Natural object and displays the corresponding source in a source window. When the source is not available, the window remains empty. The trace bar is placed on the first executable source code line.

When a user ’steps’ into a new object or when a breakpoint or watchpoint is found inside a new object, a new source window displays this object’s source.

When a user ’steps’ into an object whose source window has already been opened, this source window is moved to the foreground.

The calling object can be identified by the cursor placed at the beginning of its source line in the original source window.

The title bar of the source window shows you the name and type of the object displayed, the menu bar shows you the available menus. In addition to the "Options", "Window" and "Help" menus, the following menus are provided:

| Menu | Explanation |
|---|---|
| **Run** | Provides various ways to continue executing the object contained in the source window; execution stops each time a breakpoint or watchpoint is found, a runtime error occurs, or the object has ended. |
| **Variable** | Provides various functions to be applied to the currently active variables. |
| **Source** | Enables you to load another program source, show the current trace bar position, or search for certain items in the current source. |
| **Calls** | Provides a list of the Natural objects most recently called. |

## Run

The "Run" menu enables you to:

- proceed with the next program step,
- return to the previous level,
- execute the current object stepping automatically into any embedded object,
- execute the current object stepping automatically over any embedded object,
- proceed until either the next breakpoint or watchpoint or the end of the object,
- proceed until the next event occurs,
- exit the Natural Debugger.

### Step Into

When you choose the "Step Into" function, the next program step is executed and the trace bar is placed on the corresponding source code line.

If this source code line invokes or includes a further Natural object, the debugger steps into this object.

When a Natural program has ended, but another Natural object is still on the Natural stack, the debugger steps into this object, too.

### Step Over

When you choose the "Step Over" function, the next program step is executed and the trace bar is placed on the corresponding source code line. This time, however, the debugger steps over any invoked or included Natural object, but stops if this object contains breakpoints or watchpoints.

### Step Return

When you choose the "Step Return" function, the debugger returns to the previous program level, but stops if it finds a breakpoint or watchpoint before this previous level is reached.

### Animated Step Into

When you choose the "Animated Step Into" function, the program is executed step by step until the end of the program. The debugger steps into any Natural object invoked or included.

### Animated Step Over

When you choose the "Animated Step Over" function, the program is executed step by step until the end of the program. The debugger steps over any invoked or included Natural object; if a breakpoint or watchpoint is set, it jumps to the corresponding statement line and continues animation.

### Go!

When you choose the "Go!" function, the object is executed until the next active breakpoint or watchpoint, and the trace bar is placed on the corresponding source code line.

### Next Event!

When you choose the "Next Event!" function, this will have the same effect as the "Go!" function in a non-event driven application. In an event-driven application, however, the object is executed until the next event is sent to the application; it stops if an active breakpoint or watchpoint occurs before the next event is sent.

## Variable

The "Variable" menu enables you to:

- display the contents of selected variables,
- modify the contents of selected variables,
- watch modifications of the contents of selected variables,
- set watchpoints for selected variables.

With the Variable Facility, you can display, modify or watch the contents of and set watchpoints for all active variables.

## Dialog Boxes

When you choose the "Display", "Modify", "Watch" or "Set Watchpoint" function, a dialog box is displayed, which shows you a list of all currently active local, global, application-independent or system variables. The following boxes are part of this dialog box:

- The "Variable" text box, which shows the currently selected variable.
- The "Line Reference" or "Context ID" box, which shows the source code line number of the variable or context variable currently contained in the "Variable" text box.
  The "Line Reference" box is only displayed if the line reference is needed to make the variable selection unambiguous. This is the case if:

  - the variable belongs to a map; then the box contains the source code line number of the corresponding RULEVAR syntax element generated by the map editor;
  - the variable is either a database variable (reporting mode only) or one of the following variables: *ISN, *COUNTER , *NUMBER; then the box contains the source code line number of the corresponding database loop or access statement;
  - the variable is defined in reporting mode, but without a DEFINE DATA statement.

  The "Context ID" box is only displayed if the variable is a context variable; then the box contains the "ctx-Id" (context ID).

- The "History List" list box ("Display" function only), which contains the most recently selected variables (up to 20) using a first-in first-out mechanism.
  The history list helps you to quickly locate a variable that has been selected before. Variables can be selected from the history list in the same way as from the variable list.

- The "Variable" list box, which contains the corresponding variable listing.

## Selecting Variables

When you select (click) a variable in the variable list of a dialog box, it is shown in the corresponding "Variable" text box.

When you double-click a variable in the variable list of a dialog box or click it once and then choose OK, a further dialog box is displayed (except with the "Watch" function).

When you double-click an array or variable group:

- the individual array or group elements are displayed in the second dialog box ("Display"),
- the array is displayed in the second dialog box for modification; groups cannot be modified ("Modify"),
- a corresponding error message is displayed ("Watchpoint").

You can also select a variable by first marking it directly in the source window and then choosing the "Display", "Modify", "Watch" or "Set Watchpoint" function respectively. Then, the "Variable" text box of the corresponding dialog box exactly shows the piece of source code you have marked, which can then be quickly modified.

## Marking Text in the Source Window

In the source window, you can mark variables or character strings for selection with either the mouse or the keyboard.

When marking text using the mouse, place the mouse pointer on the first character to be selected, drag the pointer to the last character you want to select, and release the mouse button. To cancel a selection, click anywhere in the document.

When using the keyboard to mark text, cursor movement keys are used. First place the cursor on a character by using an arrow key, then press and hold down the SHIFT key and use the following keys for text selection:

- the LEFT ARROW key to mark the area to the left of your cursor position,
- the RIGHT ARROW key to mark the area to the right of your cursor position,
- the END key to mark the area until the end of the source code line,
- the HOME key to mark the area until the beginning of the source code line.

## Display

With this function, a variable can be selected from the listing in the dialog box for display along with its current content in a second "Display Variable" dialog box, where you can choose between alphanumeric and binary representation of the variable value.

When you select an array, a handle variable or a group of variables, the individual elements and their values are listed in the second dialog box. With arrays, any variable index expression is evaluated.

The element listing can be expanded or contracted by choosing the "Expand/Contract" button. Whenever the number of array, group or dialog elements on the list exceeds a certain display limit, a "More" line appears, which can be double-clicked to display further elements. Alternatively, the "Expand" function can be used again.

A variable, array or group of variables can also be selected for display in the "Display Variable" dialog box by double-clicking it with the left mouse bottom directly in the source window.

## Modify

With this function, a variable can be selected from the listing in the dialog box for display together with its current value in a second "Modify Variable" dialog box, where its value can be modified.

If you want to modify a system variable, only system variables which can be modified are displayed in the first dialog box.

If you want to modify an array, only its name but no values are displayed in the second dialog box. The value you enter will then be valid for all array elements.

Groups of variables cannot be selected for modification.

Variables or arrays can also be selected for modification of their values in the "Modify Variable" dialog box by clicking them directly in the source window, pressing the left and then the right mouse button.

## Watch

With this function, variables, arrays or groups of variables can be selected from the listing in the dialog box for placement in the "Watch Variables" window, where they can be watched for modification of their values during the program flow.

If a variable is selected and the "Watch Variables" window is iconized, it will be opened automatically showing all currently selected variables.

## Watchpoint

With this function, single variables and individual group or array elements can be selected from the listing in the dialog box for the definition of a watchpoint in a second "Set Watchpoint" dialog box; arrays and groups of variables cannot be selected.

The second "Set Watchpoint" dialog box displays the name of the watchpoint (which corresponds to the name of the selected variable) together with its line reference (if applicable), and the names of the corresponding Natural object and library; for further information on line references, see the description of the "Line Reference" box.

**Note:**
With system variables, the corresponding watchpoint is not attached to a specific library and object; therefore, the object and library name will always be SYSTEM.

To define a watchpoint, you specify the following items in the corresponding boxes:

- the state of the watchpoint,
- a condition for the watchpoint to be activated (optional),
- the number of skips before execution of the watchpoint,
- the maximum number of executions of the watchpoint,

For further information on how to specify these items, refer to the section The "Set Watchpoint" Dialog Box of the Variable facility.

**Note:**
Watchpoints can also be set with the "Set Watchpoint" function of the Variable Facility or with the "Set" function of the "Watchpoint Maintenance" facility.

# Source

The "Source" menu enables you to:

- load another source program,
- show the current trace bar position,
- search for an item in the program currently in the source window,
- repeat the most recent search operation.

## Load

With the "Load" function, you can specify a further source program for being loaded into the source window. The "Load Source" dialog box appears, in which you specify the program name and the appropriate library name if the program is not contained in the current library (default).

You can also select a character string for being placed into the "Load Source" dialog box by marking it directly in the source window and then choosing the "Load" function.

## Trace Bar Position

With this function, you can move to the current position of the trace bar.

## Search

With the "Search" function, you can search up or down through the active window to locate each occurrence of a specified word or character string.

The "Search" dialog box appears, where you can enter the text to be located in the "Search For" text box. In addition, you can turn the "Match Upper/Lower Case" and "Whole Words Only" options on or off.

The first occurrence of the search text is highlighted (selected), whereas a message lets you know if the search text could not be found.

With the "Match Upper/Lower Case" option, you can specify whether the search operation is to look for an exact match (ON) or for the same characters only, regardless of case (OFF).

With the "Whole Words Only" option, you can specify whether the search operation is to look for occurrences that are whole words only, not part of a character string (ON), or for all occurrences of the search text, whole words and parts of a character string (OFF).

To change the direction of the search, choose the "Up" button to search upwards, to the top of the text, or the "Down" button to search downwards, to the bottom of the text; "Down" is the default.

If the search does not start at the top (or bottom) of the text, and the search text cannot be found, a dialog window appears. You can choose "Yes" to continue the search at the top (or bottom) of the text or "No" to cancel the search.

You can also select a character string to be placed into the "Search for" text box by marking it directly in the source window and then choosing the "Search" function.

## Repeat Search

With this function, you repeat the previous search operation and locate the next occurrence of the search text specified with the "Search" function.

# Calls

The "Calls" menu provides you with a list (history) of the most recently called Natural objects including copycodes and inline subroutines. Up to 20 objects can be listed; the most recently called object appears at the top of the list.

The objects list consists of the following information:

- The level of the called object without counting copycodes and inline subroutines.
- The level of the called object counting copycodes and inline subroutines.
- The name of the called object.
- The type of the called object.
- The event and control handle of the event handler to be processed (with event-driven applications only).

The information line at the bottom displays additional information on the called object:

- The name of the calling object:
  "Natural" is displayed as the calling object if the called object is the application startup program or a program activated from the Natural stack (including error transaction programs and programs activated by a RUN statement from inside the application).

- The source code line in which the object was called:
  If you select an object from the list, except with "Natural", the source of the calling program is displayed in the middle of the source window with the cursor placed at the beginning of the line in which the call occurred.

# Watch Variables Facility

The "Watch Variables" facility is primarily intended to select specific variables from a set of variables for closer and permanent observation of their values.

To invoke the "Watch Variables" facility either choose it from the "Window" menu of any menu bar or double-click the corresponding icon; the "Watch Variables" window is displayed. If a variable is selected for placement into the "Watch Variables" window and the window is still iconized, it opens automatically showing all currently selected variables.

In addition to the "Options", "Window" and "Help" menus, the menu bar of the "Watch Variables" window provides you with the "Variables" menu, which offers various functions to be applied to the variables currently placed in the "Watch Variables" window.

**The "Variables" Menu**

The "Variables" menu enables you to:

- add a new variable to be displayed in the "Watch Variables" window,
- expand or contract the selected variable group, array or handle variable,
- remove the selected variable from the "Watch Variables" window,
- display the location of the selected variable,
- remove all variables from the "Watch Variables" window,
- watch the value of a selected variable in binary or alphanumeric format,
- watch all variable values in either binary or alphanumeric format,
- exit the Natural Debugger.

## Add

When you choose the "Add" function, a dialog box is displayed, which shows you a list of all currently active local, global or application-independent variables. Variables, arrays or groups of variables can be selected from this list to be placed into the "Watch Variables" window, where they can be watched for modification of their values during the program flow.

**Note:**
Variables can also be selected to be watched in the "Watch Variables" window by using the "Watch" function of either the "Variable" menu in the source window’s menu or tool bar, or the "Variables" menu of the Variable Facility.

## Expand/Contract

The "Expand/Contract" function can only be used with arrays, variable groups or handle variables. When you select an array or a group variable marked with a "+" character, you display its individual elements (Expand); when you select an array or a group variable marked with a "-" character, you return to the display of the array or group variable only (Contract). When you expand handle variables, the individual properties dialog windows are displayed.

The same can be achieved by double-clicking arrays, groups or handle variables marked with "+" or "-" directly in the "Watch Variables" window.

Whenever the number of array, group or dialog elements exceeds a certain display limit, a "More" line appears, which can be double-clicked to display further elements. Alternatively, the "Expand" function can be used again.

## Delete

With the "Delete" function, you can delete a selected variable from the "Watch Variables" window.

## Display

With the "Display" function, you can display the name of the object and the library for which the selected variable is defined. In some cases, also the variable’s source code line number is displayed.

## Delete All

With the "Delete All" function, you can delete all variables currently contained in the "Watch Variables" window.

## Binary/Alphanumeric

With the "Binary/Alphanumeric" function, you can watch the value of a selected variable in either binary or alphanumeric format.

## All Binary/Alphanumeric

With the "All Binary/Alphanumeric" function, you can watch the values of all variables currently contained in the "Watch Variables" window in either binary or alphanumeric format.

# Variable Facility

You invoke the Variables Facility by either choosing it from the "Window" menu of any menu bar or double-clicking the corresponding icon. The "Variable Facility" window is displayed, which contains the specifications of currently active local, global, application-independent or system variables:

| Item | Shows |
|---|---|
| **Lv** | The level of the displayed variable. |
| **Tp** | The type of the displayed variable; valid types are:<br>A - for array,<br>G - for variable group,<br>H - for handle variable,<br>R - for redefined variable,<br>S - for scalar variable. |
| **Name** | The name of the displayed variable. |
| **St** | The state of the displayed variable; valid states are:<br>"+" - group, array or handle variable ("contracted" display),<br>"-" - group, array or handle variable ("expanded" display),<br>"0" - single variable or individual group or array element. |
| **Line/Ctx-Id** | The source code line reference of the displayed variable. If your variable is a context variable, the "Ctx-Id" item is displayed instead (the context ID associated to the context variable). |
| **Fmt** | The format and length of the displayed variable. |
| **Contents** | The value of the variable displayed.<br>The displayed value is updated when using, for example, the "Refresh" function of the "Variables" menu. |

In addition to the "Options", "Window" and "Help" menus, the menu bar of the Variable Facility provides you with the "Variables" Menu, which offers various functions to be applied on the variables currently placed in the "Variable Facility" window.

## "Variables" Menu

The "Variables" menu enables you to:

- search for a specific variable,
- expand or contract the selected variable group, array or handle variable,
- modify the contents of the selected variable,
- watch the contents of the selected variable,
- set a watchpoint for the selected variable,
- show all local variables contained in the current program,
- show all variables of the currently active global data area,
- show all application-independent variables,
- show all context variables,
- show all system variables,
- show all variable values in either binary or alphanumeric format,
- show the latest values of the currently displayed variables,
- exit the Natural Debugger.

## Search

With the "Search" function, you can search up or down through the active window to locate each occurrence of a specified name or character string.

The "Search" dialog box appears, where you can enter the text to be located in the "Search For" text box. In addition, you can turn the "Match Upper/Lower Case" and "Whole Words Only" options on or off.

The line with the first occurrence of the search text is highlighted (selected), whereas a message lets you know if the search text could not be found.

With the "Match Upper/Lower Case" option, you can specify whether the search operation is to look for an exact match (ON) or for the same characters only, regardless of case (OFF).

With the "Whole Words Only" option, you can specify whether the search operation is to look for occurrences that are whole words only, not part of a character string (ON), or for all occurrences of the search text, whole words and parts of a character string (OFF).

To change the direction of the search, choose the "Up" button to search upwards, to the top of the text, or the "Down" button to search downwards, to the bottom of the text; "Down" is the default. If the search does not start at the top (or bottom) of the text, and the search text cannot be found, a dialog window appears. You can choose "Yes" to continue the search at the top (or bottom) of the text or "No" to cancel the search.

## Repeat Search

With this function you repeat the previous search operation and locate the next occurrence of the search text specified with the "Search" function.

## Expand/Contract

The "Expand/Contract" function can only be used with arrays, variable groups or handle variables. When you select an array or a group variable with a "+" character in the "St" column, you can display its individual elements (Expand); when you select an array or a group variable with a "-" character in the "St" column, you return to the display of the array or group variable only. When you expand handle variables, the individual properties dialog windows are displayed.

The same can be achieved by double-clicking arrays, groups or handle variables with state "+" or "-" directly in the "Variable Facilities" window.

Whenever the number of array, group or dialog window elements exceeds a certain display limit, a "More" line appears, which can be double-clicked to display further elements. Alternatively, the "Expand" function can be used again.

## Modify

With the "Modify" function, a variable or array can be selected from the "Variable Facility" window for modification of its value in the "Modify Variable" dialog box.

The new value must be valid for the format of the variable or array.

When you select an array, only its name but no values are displayed in the dialog box. The value you enter will then be valid for all array elements.

Groups of variables cannot be selected for modification.

**Note:**
Variables or arrays can also be selected for modification of their values in the "Modify Variable" dialog box by using the "Modify" function of the "Variable" menu in the source window’s menu or tool bar.

## Watch

With the "Watch" function, variables, arrays and groups of variables can be placed into the "Watch Variables" window, where they can be watched for modification of their values during the program flow.

If a variable is selected and the "Watch Variables" window is iconized, it will be opened automatically showing all currently selected variables.

**Note:**
Variables can also be selected to be watched in the "Watch Variables" window by using either the "Watch" function of the "Variable" menu in the source window’s menu or tool bar, or the "Watch Variables" function itself.

## Set Watchpoint

With the "Set Watchpoint" function, single variables and individual group or array elements can be selected from the "Variable Facility" window for the definition of a watchpoint in the "Set Watchpoint" dialog box; arrays and groups of variables cannot be selected.

## "Set Watchpoint" Dialog Box

The "Set Watchpoint" dialog box displays the name of the watchpoint (which corresponds to the name of the selected variable) together with its line reference/context ID (if applicable) and the names of the corresponding Natural object and library; for further information on line references/context IDs, see the description of the "Line Reference" box.

**Note:**
With system variables, the corresponding watchpoint is not attached to a specific library and object; therefore, the object and library name will always be SYSTEM.

To define a watchpoint, you have to specify the following items in the corresponding boxes:

- The state of the watchpoint to be set; valid states are "active" (default) and "pending".
- A condition for the watchpoint to be activated (optional).

You can specify an appropriate value and watchpoint operator; if no operator and value (that is, condition) is specified, the default setting (MOD) applies (for a description of the individual watchpoint operators, see below).

- The number of skips before execution of the watchpoint if it is not to be executed until the program has run a certain number of times; the default is "0".
- The maximum number of executions of the watchpoint; the default is "0".

Watchpoint specifications are not set until you either choose the "Set" button or press ENTER. If you choose the "Cancel" button or press ESC, no watchpoint will be set.

Once a watchpoint has been specified, it remains set until you either leave your Natural application or explicitly delete it.

# Watchpoint Operators

Watchpoint operators are set via option buttons; the available watchpoint operators are:

| Operator | Explanation |
|---|---|
| **MOD** | **Modification**<br>This is the default setting. It indicates that the watchpoint is activated each time a modification of the variable occurs. |
| **EQ** | **Equal**<br>The EQ operator indicates that the watchpoint is activated only when the current value of the variable is equal to the specified value. |
| **NE** | **Not Equal**<br>The NE operator indicates that the watchpoint is activated only when the current value of the variable is not equal to the specified value. |
| **GT** | **Greater Than**<br>The GT operator indicates that the watchpoint is activated only when the current value of the variable is greater than the specified value. |
| **GE** | **Greater or Equal**<br>The GE operator indicates that the watchpoint is activated only when the current value of the variable is greater than or equal to the specified value. |
| **LT** | **Less Than**<br>The LT operator indicates that the watchpoint is activated only when the current value of the variable is less than the specified value. |
| **LE** | **Less or Equal**<br>The LE operator indicates that the watchpoint is activated only when the current value of the variable is less than or equal to the specified value. |

**Note:**
Watchpoints can also be set with the "Watchpoint" function in the source window or with the "Set" function of the "Watchpoint Maintenance" facility.

## Local

With the "Local" function, you can display all local variables of the current program in the "Variable Facilities" window.

With this function, also internally generated temporary variables are displayed, which, in case of system variable access, have the same name as the system variables (for example, *EVENT), from which they receive their values.

## GDA

With the GDA function, you can display all variables of the currently active global data area (GDA) in the "Variable Facilities" window.

## AIV

With the AIV function, you can display all application-independent variables (AIVs) in the "Variable Facilities" window.

### CTX

With the CTX function, you can display all content variables in the "Variable Facilities" window.

### System

With the "System" function, you can display all Natural system variables in the "Variable Facilities" window.

### All Binary/Alphanumeric

With the "All Binary/Alphanumeric" function, you can switch between binary and alphanumeric representation of the values of all variables displayed in the "Variable Facilities" window.

## Refresh

With the "Refresh" function, you can show the latest values of the variables currently displayed in the "Variable Facilities" window.

Since an automatic refresh of variable values would be too time-consuming here, latest variable values are automatically displayed with the "Watch Variables" facility only.

# Watchpoint Maintenance

Using watchpoints, you can rapidly detect "illegal" alterations to Natural variables by objects that contain errors.

By default, watchpoints are used to instruct the Natural Debugger to interrupt the execution of a Natural object when the contents of a variable change. However, by specifying a certain value for the variable together with a watchpoint operator when setting a watchpoint, a condition can be set which only activates the watchpoint when it becomes true.

A variable is considered to have changed either when its current value differs from the value recorded when the watchpoint was last triggered or when it differs from the initial value.

A watchpoint is defined and maintained with the "Watchpoint Maintenance" facility. You invoke the "Watchpoint Maintenance" facility by either choosing it from the "Window" menu of any menu bar or double-clicking the corresponding icon. The "Watchpoint Maintenance" window is displayed, which contains a list of all watchpoints defined for the current program.

The following items are listed:

| Item | Shows |
|---|---|
| **No** | The number of the listed watchpoint. |
| **Variable** | The name of the variable for which the watchpoint was set. |
| **Object** | The name of the object which contains the listed watchpoint. |
| **Count** | Shows how often a watchpoint has been executed. |

In addition to the "Options", "Window" and "Help" menus, the menu bar of the "Watchpoint Maintenance" facility contains the "Watchpoint" menu, which provides you with the possibility to define new watchpoints and with various functions to be applied to the watchpoints currently listed in the "Watchpoint Maintenance" window.

## "Watchpoint" Menu

The "Watchpoint" menu enables you to:

- define a new watchpoint,
- activate or deactivate a selected watchpoint,
- delete a selected watchpoint,
- display a selected watchpoint,
- modify a selected watchpoint,
- delete all watchpoints set in the current program,
- deactivate all watchpoints in the current program,
- activate all watchpoints in the current program.

Some of the provided functions require to previously select a watchpoint. To select a watchpoint in the "Watchpoint Maintenance" window, click it with the left mouse button.

## Set

With this function, single variables, arrays and individual group elements can be selected from the listing in the "Set Watchpoint" dialog box for the definition of a watchpoint in a second "Set Watchpoint" dialog box; entire groups of variables cannot be selected.

The second "Set Watchpoint" dialog box displays the name of the watchpoint (which corresponds to the name of the selected variable) together with its line reference/context ID (if applicable) and the names of the corresponding Natural object and library; for further information on line references/context IDs, see the description of the "Line Reference" box .

**Note:**
With system variables, the corresponding watchpoint is not attached to a specific library and object; therefore, the object and library name will always be SYSTEM.

To define a watchpoint, you have to specify the following items in the corresponding boxes:

- the state of the watchpoint,
- a condition for the watchpoint to be activated (optional),
- the number of skips before execution of the watchpoint,
- the maximum number of executions of the watchpoint,

For further information on how to specify these items, refer to the section "Set Watchpoint" Dialog Box of the Variable Facility.

**Note:**
Watchpoints can also be set with the "Watchpoint" function in the source window or with the "Set Watchpoint" function of the Variable Facility.

## Activate/Deactivate

With the "Activate/Deactivate" function, you can set the current state of the selected watchpoint to "active" or "pending" respectively.

## Delete

With the "Delete" function, you can delete the selected watchpoint.

## Display

With the "Display" function, you can display the selected watchpoint. The "Display Watchpoint" dialog box appears showing you all the specifications of this watchpoint. These are:

- The number of the listed watchpoint.
- The state of the listed watchpoint; valid states are "active" (default) or "pending".
- The name of the variable for which the watchpoint was set.
- The source code line reference of the variable for which the watchpoint is set/the context ID of the context variable (if either is applicable); for further information on line references/context variables, see the description of the "Line Reference" box.
- The operator which has been set for the listed watchpoint; the default operator is MOD.
- The value specified with the watchpoint operator; with the default operator (MOD), no value (condition) can be specified.
- The name of the object which contains the listed watchpoint.
- The library which contains the object with the listed watchpoint.
- The number of skips before execution of the watchpoint.
- The maximum number of executions of the watchpoint.
- The number of how often a watchpoint has been executed.

## Modify

With the "Modify" function, you can modify the selected watchpoint. The "Modify Watchpoint" dialog box appears, where you can modify the specifications of this watchpoint.

For further information on these items, refer to the section "Set Watchpoint" Dialog Box of the Variable Facility.

Any modifications to the watchpoint are not performed until you choose the "Modify" button or press ENTER. If you choose the "Cancel" button or press ESC, the watchpoint remains unchanged.

You can also select a watchpoint for being modified in the "Modify Watchpoint" dialog box by double-clicking it directly in the "Watchpoint Maintenance" window.

## Delete All

With the "Delete All" function, you can delete all watchpoints currently listed in the "Watchpoint Maintenance" window.

## Deactivate All

With the "Deactivate All" function, you can deactivate all watchpoints currently listed in the "Watchpoint Maintenance" window.

## Activate All

With the "Activate All" function, you can activate all watchpoints currently listed in the "Watchpoint Maintenance" window.

# Breakpoint Maintenance

A breakpoint is a point at which control is returned to the user while a Natural object is executing.

Breakpoints are defined and maintained with the "Breakpoint Maintenance" facility. You invoke the "Breakpoint Maintenance" facility by either choosing it from the "Window" menu of any menu bar or double-clicking the corresponding icon. The "Breakpoint Maintenance" window is displayed, which contains a list of all breakpoints defined for the current program.

The following items are listed:

| Item | Explanation |
|---|---|
| **No** | The number of the listed breakpoint. |
| **Line** | The number of the source code line on which the breakpoint is set. |
| **Object** | The name of the object which contains the listed breakpoint. |
| **Count** | Displays how often a breakpoint has been executed. |

In addition to the "Options", "Window" and "Help" menus, the menu bar of the "Breakpoint Maintenance" facility contains the "Breakpoint" menu, which provides you with both the possibility to define new breakpoints and various functions to be applied to the breakpoints currently listed in the "Breakpoint Maintenance" window.

## "Breakpoint" Menu

The "Breakpoint" menu enables you to:

- define a breakpoint,
- activate or deactivate a selected breakpoint,
- delete a selected breakpoint,
- display a selected breakpoint,
- modify a selected breakpoint,
- delete all breakpoints set in the current program,
- deactivate all breakpoints in the current program,
- activate all breakpoints in the current program.

Some of the provided functions require to previously select a breakpoint. To select a breakpoint in the "Breakpoint Maintenance" window, click it with the left mouse button.

## Set

With the "Set" function, you can define a new breakpoint. The "Set Breakpoint" dialog box is displayed, where you define the breakpoint by specifying the following items in the corresponding boxes:

- The state of the breakpoint to be set; valid states are "active" (default) and "pending".
- The name of the Natural object to contain the breakpoint; the default object name is the name of the object currently in the source window.
- The name of the Natural library that contains the object with the breakpoint; the default library name is the name of the library which contains the object currently in the source window.
- The line number of the object’s source code where the breakpoint is to be executed.

"Begin" means that the breakpoint is to be set at the first executable line of code of the specified object; "End" means that the breakpoint is to be set at the last executable line of code of the specified object.

- The number of skips before execution of the breakpoint if it is not to be executed until the program has run a certain number of times; the default is "0".
- The maximum number of executions of the breakpoint; the default is "0".

Breakpoint specifications are not set until you either choose the "Set" button or press ENTER. If you choose the "Cancel" button or press ESC, no breakpoint will be set.

Breakpoints can also be set directly in the program currently contained in the source window by double-clicking the appropriate statement line with the right mouse button. This way, a breakpoint is defined with all default values and the corresponding source code line number. It can be displayed and/or modified by using the corresponding functions.

Breakpoints cannot be set on comment lines, or on any statement line other than the first one if a single statement occupies more than one line.

Once a breakpoint has been defined, it remains set until you either leave your Natural application or explicitly delete it.

## Activate/Deactivate

With the "Activate/Deactivate" function, you can set the current state of the selected breakpoint to "active" or "pending" respectively.

## Delete

With the "Delete" function, you can delete the selected breakpoint.

A breakpoint can also be deleted directly in the program currently contained in the source window by double-clicking it with the right mouse button.

## Display

With the "Display" function, you can display the selected breakpoint. The "Display Breakpoint" dialog box appears showing you all the specifications of this breakpoint. These are:

- The number of the listed breakpoint.
- The state of the listed breakpoint; valid states are "active" (default) or "pending".
- The name of the object which contains the listed breakpoint.
- The library which contains the object with the listed breakpoint.
- The number of the source code line on which the breakpoint is set.
- The number of skips before execution of the breakpoint.
- The maximum number of executions of the breakpoint.
- The number of how often a breakpoint has been executed.

## Modify

With the "Modify" function, you can modify the selected breakpoint. The "Modify Breakpoint" dialog box appears, where you can modify the specifications of this breakpoint, see also section Set.

Any modifications to the breakpoint are not performed until you either choose the "Modify" button or press ENTER. If you choose the "Cancel" button or press ESC, the breakpoint remains unchanged.

You can also select a breakpoint for being modified in the "Modify Breakpoint" dialog box by double-clicking it directly in the "Breakpoint Maintenance" window.

## Delete All

With the "Delete All" function, you can delete all breakpoints currently listed in the "Breakpoint Maintenance" window.

## Deactivate All

With the "Deactivate All" function, you can deactivate all breakpoints currently listed in the "Breakpoint Maintenance" window.

## Activate All

With the "Activate All" function, you can activate all breakpoints currently listed in the "Breakpoint Maintenance" window.